Introduction

Text embeddings are learned vector representation of text in which similar words/ sentences have a similar encoding. It is the proxy for a wide collection of machine learning models to understand text information. Text embeddings can be trained using various models, corpora, initializations, etc. In this project we ask:

Do different embeddings capture the same information?
What are the similarities and differences between embeddings learnt with different models, datasets, and initializations?

Figure 1. Text embeddings encode various linguistic information.

Objectives

Get Different Embeddings
Train/Inference different word/sentence embedding models using different corpus and from different initializations.

Embedding Alignment
Find a reasonable alignment mechanism that can deal with embeddings of distinctive dimensions, vocabulary size, etc.

Difference Correlation
Determine whether there is correlation between embedding similarities and their downstream performances.

Embedding Ensemble
Ensemble word/sentence embeddings or outputs of models that take these embeddings as inputs for better performance.

Preliminaries and Methods

Alignment and Matching of Text Embeddings
Embeddings that capture the same information may appear to be completely different; to correctly find similar embeddings, we can align embeddings before comparing them, as text embeddings trained by most used algorithms are rotationally invariant.

Figure 2. Equivalent but “Different” Embeddings

Figure 3. Demonstration of Alignment Between Two Different Embeddings

Table 1. Alignment results between different pairs of embeddings (vocab size ~ 5000)

<table>
<thead>
<tr>
<th>Metric</th>
<th>C1-C2</th>
<th>G1-G2</th>
<th>C1-G1</th>
<th>LSTM-C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. $d$, distance (Least Squares)</td>
<td>0.309</td>
<td>0.623</td>
<td>0.965</td>
<td>0.937</td>
</tr>
<tr>
<td>Matching Acc. (nearest, top1)</td>
<td>1.000</td>
<td>0.634</td>
<td>0.4302</td>
<td>0.2284</td>
</tr>
<tr>
<td>Matching Acc. (nearest, top5)</td>
<td>1.000</td>
<td>0.8769</td>
<td>0.5712</td>
<td>0.4388</td>
</tr>
<tr>
<td>Matching Acc. (OT)</td>
<td>1.000</td>
<td>1.000</td>
<td>0.8842</td>
<td>0.6328</td>
</tr>
</tbody>
</table>

* C1 and C2 refer to Word2Vec (CBOW)\(^1\) embeddings trained with different initializations; G1 and G2 refer to GloVe\(^2\) embeddings trained with different initializations; all trained on IMDB\(^3\) dataset. LSTM refers to pretrained Word2Vec embeddings fine-tuned by LSTM model\(^4\) on sentiment analysis.

Figure 4. Illustration of Different Ensembling Methods

Figure 5. Correlation of Embedding Similarity and Downstream Performance on Classification Tasks. Each point corresponds to a pair of embeddings.

Figure 6. Accuracy Improvement on IMDB Sentiment Analysis After Ensembling (The GloVe embeddings are pretrained on WikiText; + GigaWord corpora\(^6\) (6 billion tokens), and the other embeddings are pretrained on IMDB review corpora\(^7\) (5 million tokens).)

Results

Embedding Alignment and Optimal Transport Matching

Optimal Transport Matching
To evaluate the similarities of embeddings $A$ and $B$ after alignment, we used the matching induced by the earth mover optimal transport allocation.

With cosine similarity as the distance metric, we consider the matching induced by the doubly stochastic matrix $\Gamma \in \mathbb{R}^{n \times n}$, that solves:

$$\sum_{i,j} \Gamma_{ij} \cos(A_i, B_j)$$

When no weighting is applied, this linear program returns a permutation matrix from which we can directly infer the matching.

Word Embedding Ensemble
Previous works have shown that in vision classification tasks, due to the “multi-view” property\(^1\) of the input data one can improve model accuracy by averaging the logit outputs of different deep learning models.

Likewise, we propose three ways of ensembling-like methods.

Conclusions

• Embeddings trained on different corpora from different models and random initializations are different given the large geometric distances between these embeddings; nevertheless, they captured similar features of the input text data since optimal transport can find a reasonable matching between the aligned embeddings.

• For simple classification tasks such as sentiment analysis and news headline classification, dissimilar embeddings have different downstream performance, but it is inconclusive whether similar embeddings have similar downstream performance.

• For classification tasks,ensembling embeddings or their outputs can improve downstream performance, while the latter leads to a higher improvement in accuracy, the former provides insights for distilling better embeddings.

References